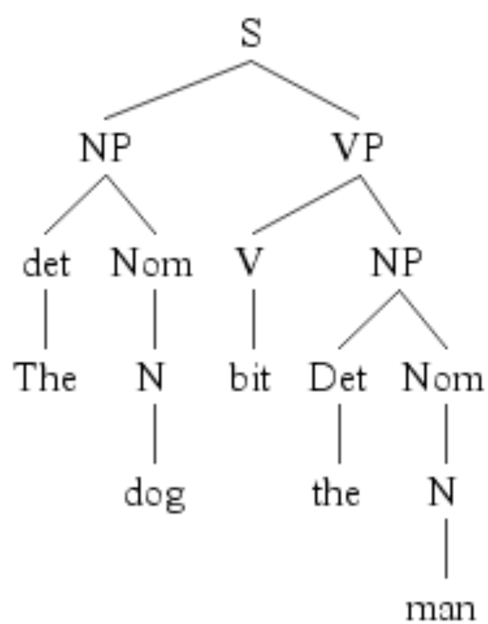
Syntactic Parsing
(15 Apr 2015)

Structural Descriptions

- Tree



- Bracketed Structure

[[The [dog]] [bit [the [man]]]]

- Labeled bracketed structure

[S [NP [det The] [Nom [N dog]]] [VP [V bit] [NP [Det the] [Nom [N man]]]]]

Context Free Grammar

$G = \langle N, \Sigma, P, S \rangle$ where:

- N is a set of non-terminal symbols, typically S,A,B,...
- S is the starting or goal symbol from N, i.e., $S \in N$
- Σ is a set of terminal symbols, typically x, y, z, . . . disjoint from N
- P is a set of production rules of the form $A \rightarrow \beta$, where:
  - A is a non-terminal $A \in N$
  - β is a string of symbols from $(\Sigma \cup N)$

CFGs for Natural Language

- A nonterminal symbol labels a syntactic part (constituent):
  NP, *VP*, *PP*, (*Noun*, *Verb*, *Det*)

- A starting symbol indicates which symbol has to come first; it labels the largest constituent or biggest part:
  S, *ROOT*, or *TOP*

- A terminal symbol labels the smallest part, the actual strings of the language:
  man, *they*, *swim*

CFGs for Natural Language

- Production rule (re-write rule): one symbol is rewritten (→) as one or more others: NP → Det Noun
- A production rule captures the notion of syntactic constituency.
- 'LHS' is used to indicate the left-hand side of the → , and likewise for 'RHS'.

Rules in Treebanks

- Lots of them! 17,000 in PTB
  - Most very flat
  - Many tailored to single sentences
  - Number grows linearly with corpus
- Largest number: S, NP, VP

Questions for Parsing

- Is this sentence in the language?
  - FSAs accept the regular languages defined by automaton
  - Parsers accept language defined by CFG
- What is the syntactic structure of this sentence?
  - Syntactic parse provides framework for semantic analysis
    - What is the subject?
    - Useful for e.g. question answering

Parsing as Search

- Search through possible parse trees
- Want one (or more) that derive input
- Formally, search problems are defined by:
  - Start state S,
  - Goal state G,
  - Successor Function:
    Transitions between states,
  - Path cost function

Parse Search Strategies

- Two constraints:
  - Must start with the start symbol
  - Must cover exactly the input string
- Correspond to main parsing search strategies
  - Top-down search (Goal-directed search)
  - Bottom-up search (Data-driven search)

Parse Search Strategies

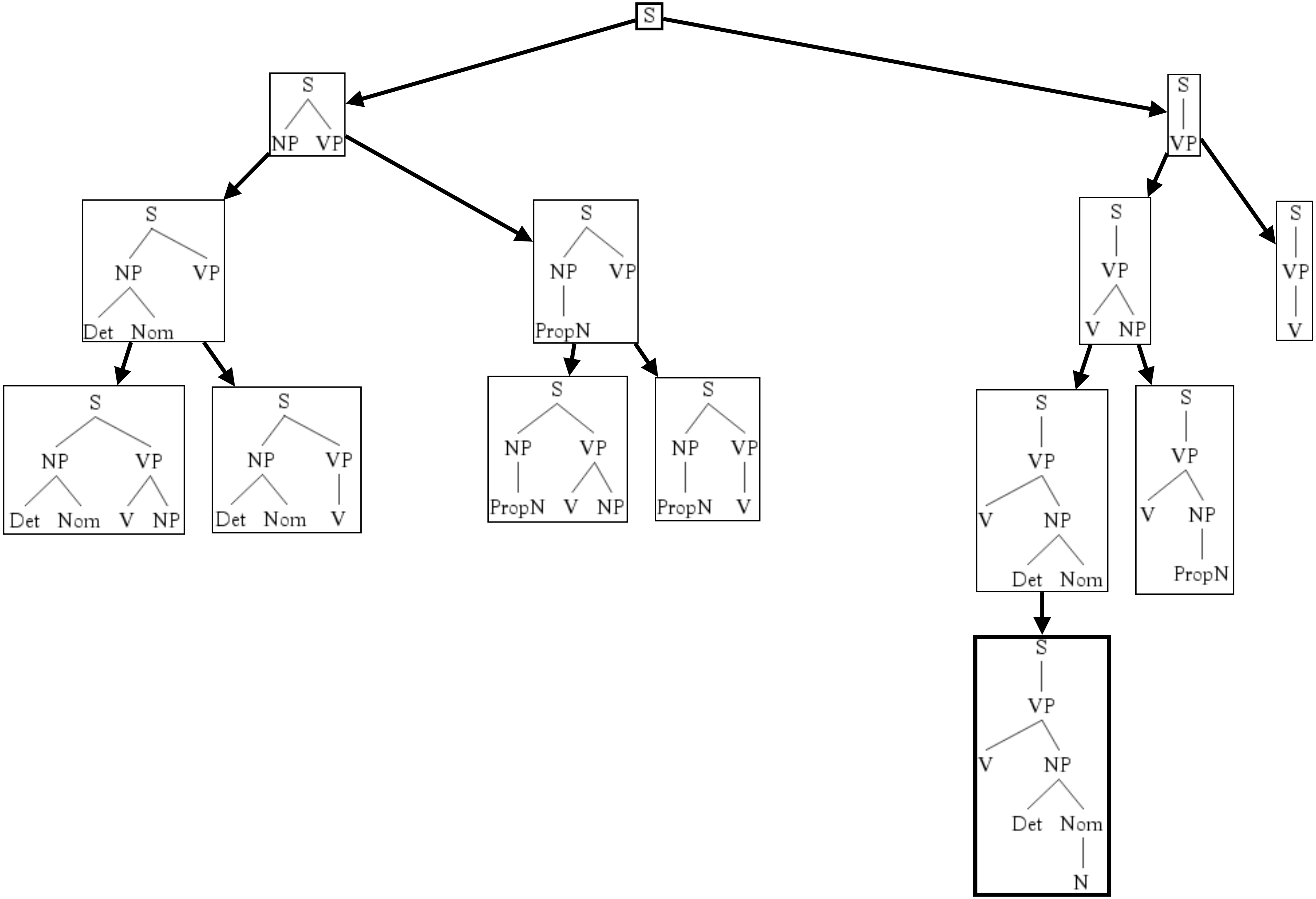
	Breadth-First	Depth-First
Top-Down		
Bottom-Up		

A Toy Grammar

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Top-Down Search

- Begin with productions with S on LHS
  - E.g., S → NP VP
- Successively expand non-terminals
  - E.g., NP → Det Nominal; VP → V NP
- Terminate when all leaves are terminals
  - *Book that flight*

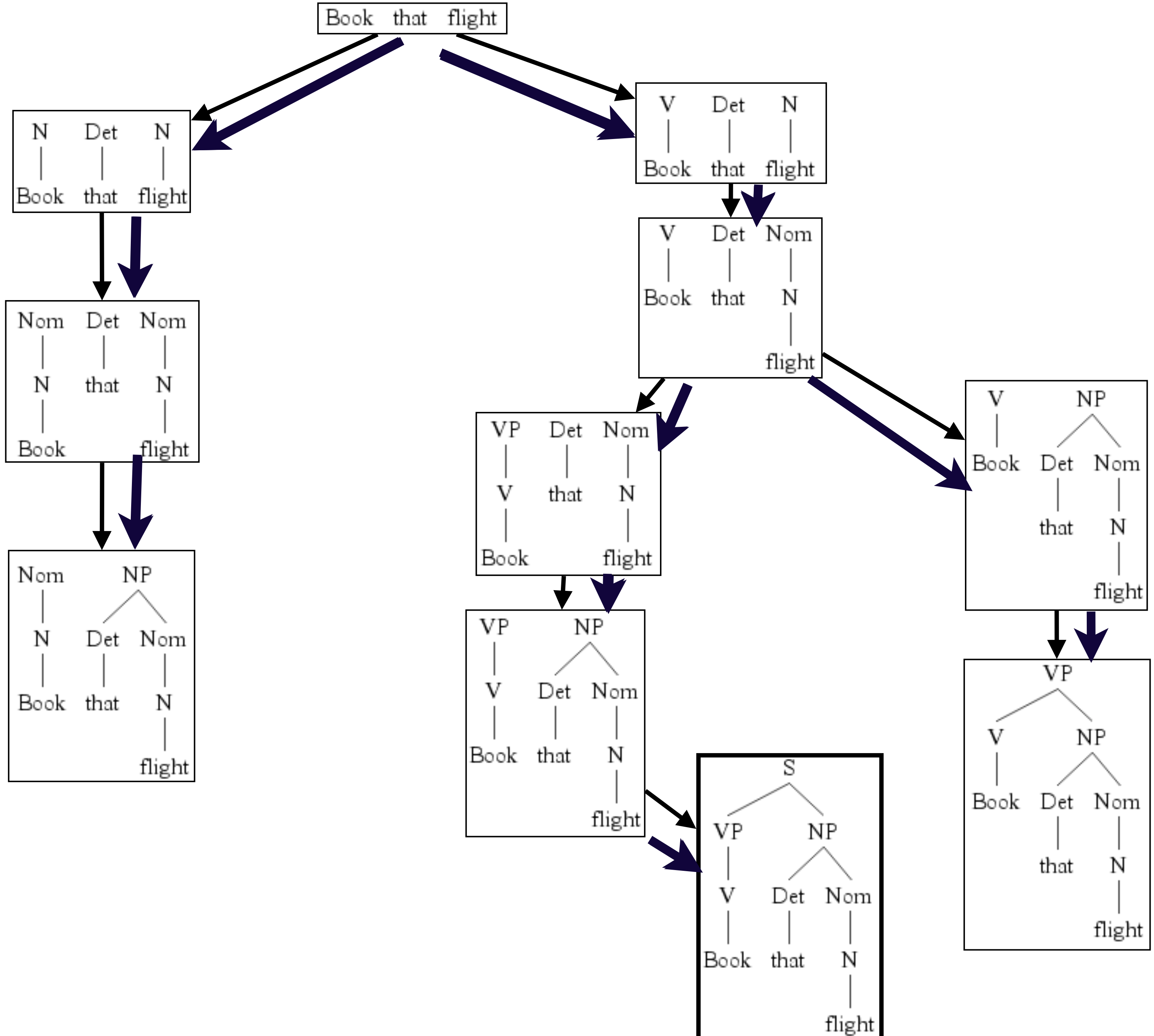

S
S NP VP
S VP
S NP VP Det Nom
S NP VP PropN
S VP V NP
S VP V
S NP VP Det Nom V NP
S NP VP Det Nom V
S NP VP PropN V NP
S NP VP PropN V
S VP V NP Det Nom
S VP V NP PropN
S VP V NP Det Nom N

Pros and Cons of Top-Down Search

- Pros:
  - Doesn't explore trees not rooted at S
  - Doesn't explore invalid subtrees
- Cons:
  - Produces trees that may not match input
  - May not terminate with recursive rules
  - May rederive subtrees during search

Bottom-Up Search

- Find all trees that span the input
  - Start with input string: *Book that flight.*
- Use all productions with current subtree(s) on RHS
  - E.g., N → Book; V → Book
- Stop when spanned by S (or no more rules apply)


Book that flight
N Det N
Book that flight
V Det N
Book that flight
Nom Det Nom
N that N
Book flight
V Det Nom
Book that N
flight
Nom NP
N Det Nom
Book that N
flight
VP Det Nom
V that N
Book flight
V NP
Book Det Nom
that N
flight
VP NP
V Det Nom
Book that N
flight
S
VP NP
V Det Nom
Book that N
flight
VP
V NP
Book Det Nom
that N
flight

Pros and Cons of Bottom-Up Search

- Pros:
  - Only explore trees that match input
  - Fewer problems with recursive rules
  - Useful for incremental/ fragment parsing
- Cons:
  - Explore subtrees that will not fit full sentences

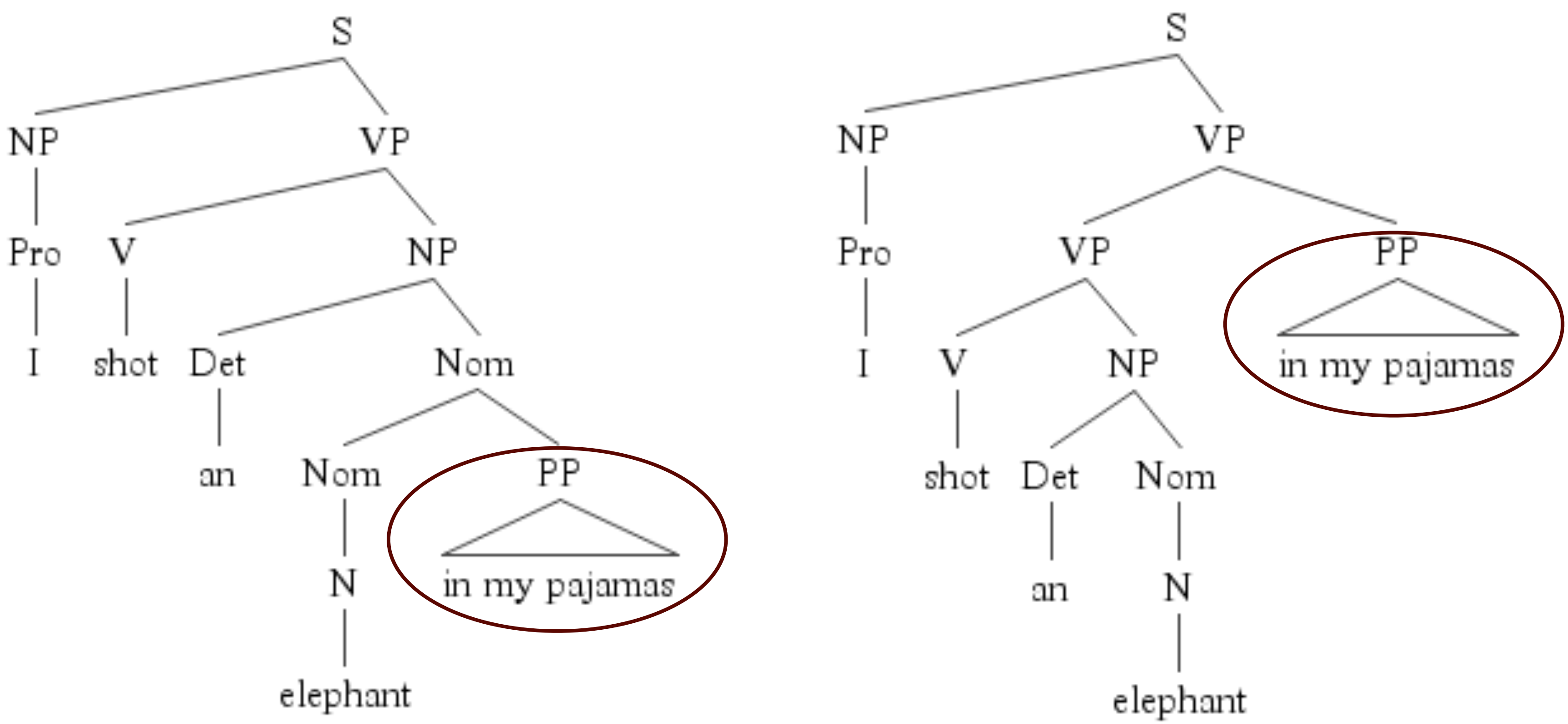
Parsing Challenges

- Ambiguity
- Recursion
- Repeated substructure

Parsing Ambiguity

- Lexical ambiguity
  - Book/N; Book/V
- Structural ambiguity:
  - Attachment ambiguity
    - Constituent can attach in multiple places
    - I shot an elephant in my pyjamas.
  - Coordination ambiguity
    - Different constituents can be conjoined
    - Old men and women

Attachment Ambiguity



Disambiguation

- Local ambiguity:
  - Ambiguity in subtree, resolved globally
- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
  - Alternatively, keep all

Resolving Global Ambiguity

Exploit other information

- Statistical
  - Some prepositional structs more likely to attach high/low
  - Some phrases more likely, e.g., (old (men and women))
- Semantic
- Pragmatic (e.g., elephants and pyjamas)

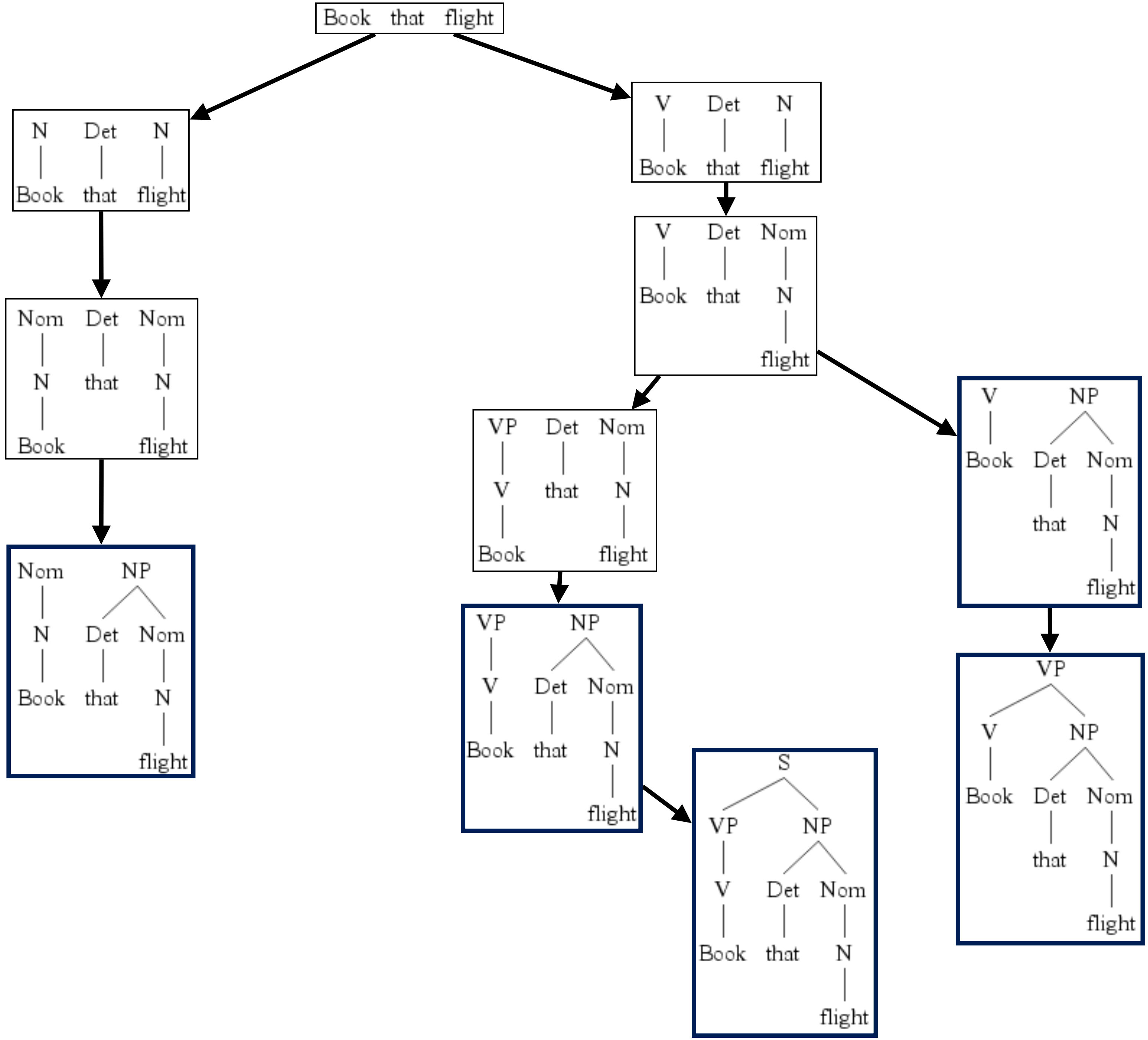
Recursion

- Direct Recursion (e.g., S → S CONJ S)
  - *water under the bridge, Bill ran and Jane jogged*
- Indirect Recursion
  - *. . . on a thimble in a box on a stool beside a table near a sofa ...*

    NP → DT Nom
    Nom → Nom PP
    PP → Prep NP
- Can yield infinite searches
  e.g., Top-down search with S → S conj S

Repeated Work

- Avoid repeatedly parsing substructures
  - Good subtrees in globally bad parses
  - Overall, bad parses will fail
  - Reconstruction subtrees on other branches
  - Can't avoid with static backtracking
- Store shared substructure for efficiency
  - Typically with dynamic programming



Book that flight
N Det N
Book that flight
V Det N
Book that flight
V Det Nom
Book that N
flight
Nom Det Nom
N that N
Book flight
VP Det Nom
V that N
Book flight
V NP
Book Det Nom
that N
flight
Nom NP
N Det Nom
Book that N
flight
VP NP
V Det Nom
Book that N
flight
S
VP NP
V Det Nom
Book that N
flight
VP
V NP
Book Det Nom
that N
flight

Parsing w/Dynamic Programming

- Makes parsing algorithms (relatively) efficient
  - Polynomial time in input length
  - Typically cubic (n^3) or less
- Several different implementations
  - Cocke-Kasami-Younger (CKY) algorithm
  - Earley algorithm
  - Chart parsing

Chomsky Normal Form (CNF)

- CKY parsing requires grammars in CNF
- All productions of the form:
  - $A \rightarrow B\ C$, or
  - $A \rightarrow a$
- Most of our grammars are not of this form
  E.g., S -> Wh-NP Aux NP VP
- Need a general conversion procedue

Hybrid Rule Conversion

- Replace all terminals with dummy non-terminals
  - Problem Rule: INF-VP → to VP
  - New Rules:
    - INF-VP → TO VP
    - TO → to

Long Productions Conversion

- Introduce new non-terminals and spread over rules
- Old Rule: S → Aux NP VP
- New Rules:
  S → X_1 VP
  X_1 → Aux NP

Result of CNF Conversion

$\mathscr{L}_1$ **Grammar**	$\mathscr{L}_1$ **in CNF**
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

Grammatical Equivalence

- Weak equivalence:
  - Recognizes same language
  - Yields different structure
- Strong equivalence
  - Recognizes same languages
  - Yields same structure
- CNF is weakly equivalent

CKY Algorithm

- Bottom-up parsing algorithm
- (Relatively) efficient
- Tabulate substring parses to avoid repeated work

CKY Approach

- Use a CNF grammar
- Build an (n+1) x (n+1) matrix to store subtrees
- Use Upper triangular portion
- Incrementally build parse spanning whole input string

Dynamic Programming in CKY

- For a parse spanning substring [i,j],
  - There must be parses spanning [*i*,*k*] and [*k*,*j*] for some *k*.
- Construct parses for whole sentence by building up from stored partial parses
- To have A → B C in [i,j],
  - We must have B in [i,k] and C in [k,j], for some i<k<j
- CNF grammar forces this for all j>i+1

CKY Approach

- Given an input string S of length n,
  - Build table (n+1) x (n+1)
  - Indexes MATCH inter-word positions:
    $_0$ Book $_1$ That $_2$ Flight $_3$
  - Cell [i,j] contains all constituents spanning (i,j)
  - [j-1,j] contains pre-terminals
  - If [0,n] contains START, the input is recognized

Chart Filling Order

- Table fills:
  - Column-by-column
  - Left-to-right
  - Bottom-to-top

- Why?
  - Necessary info available (below and left)
  - Allows online sentence analysis
  - Works across input string as it arrives

Is this a Parser?

Is this a Parser?

- Sort of...
- It's a recognizer.
- What if we want the actual parses?

CKY Example

Book	the	flight	through	Houston
S, VP, Verb, Nominal, Noun [0,1]	--- [0,2]	S, VP, X2 [0,3]	--- [0,4]	S, VP, X2, X2, S, VP, [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	--- [2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, ProperNoun [4,5]

X2 → X2 PP

S → X2 PP

VP → X2 PP

VP → VP PP

Learning Probabilities

- Simplest: Treebank of parsed sentences
  - To compute probability of a rule, count:
    - Times LHS is expanded
    - Times LHS expands to RHS

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

Example PCFG

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that\ [.10] \mid a\ [.30] \mid the\ [.60]$
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book\ [.10] \mid flight\ [.30]$
$S \rightarrow VP$	[.05]	$\mid meal\ [.15] \mid money\ [.05]$
$NP \rightarrow Pronoun$	[.35]	$\mid flights\ [.40] \mid dinner\ [.10]$
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book\ [.30] \mid include\ [.30]$
$NP \rightarrow Det\ Nominal$	[.20]	$\mid prefer;\ [.40]$
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I\ [.40] \mid she\ [.05]$
$Nominal \rightarrow Noun$	[.75]	$\mid me\ [.15] \mid you\ [.40]$
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston\ [.60]$
$Nominal \rightarrow Nominal\ PP$	[.05]	$\mid NWA\ [.40]$
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does\ [.60] \mid can\ [40]$
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from\ [.30] \mid to\ [.30]$
$VP \rightarrow Verb\ NP\ PP$	[.10]	$\mid on\ [.20] \mid near\ [.15]$
$VP \rightarrow Verb\ PP$	[.15]	$\mid through\ [.05]$
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	